Corrigé DS2

Problème 1

1. Ce terme est un (string, char) terme qui représente le terme $x + \cos(y)$.

2. Il suffit de vérifier pour chacun des symboles de fonctions si la taille de la liste de ses arguments est égale à son arité. On peut exploiter pour ce faire List.for_all :

```
let rec terme_bien_forme t :bool = match t with
  |V(_) -> true
  |F((_, arite), l) -> (List.length l = arite) && List.for_all terme_bien_forme l
```

3. On utilise le terme défini en question 1 :

```
let f = Forall('x', R(("egal",2),[t;t]))
```

4. Même principe que pour la question 3. On prend garde de vérifier la correction des arités des fonctions utilisées dans chacun des termes dans le cas d'une formule atomique :

```
let rec formule_bien_formee f = match f with
  |R((_,arite),l) -> List.length l = arite && List.for_all terme_bien_forme l
  |Forall(_,g) |Exists(_,g) -> formule_bien_formee g
  |Non(g) -> formule_bien_formee g
  |Op_bin(_,g,h) -> (formule_bien_formee g) && (formule_bien_formee h)
```

5. On peut exploiter List.map conjointement avec List.mem ou utiliser directement List.exists :

```
let rec apparait x t = match t with
  |V(y) -> x=y
  |F(_,l) -> List.mem true (List.map (apparait x) l)
(* ou : List.exists (apparait x) l *)
```

6. Une occurrence de variable est libre si elle n'est sous la portée d'aucun quantificateur. On obtient :

```
let rec est_libre x f = match f with
  |R(_,l) -> List.exists (apparait x) l
  |Forall(y,g) |Exists(y,g) -> x<>y && (est_libre x g)
  |Non(g) -> est_libre x g
  |Op_bin(_,g,h) -> est_libre x g || est_libre x h
```

7. Une formule est close si toutes les occurrences de chacun de ses variables est liée. On applique donc pour chacune des variables v le traitement suivant : vérifier si une occurence de v est libre à l'aide de est_libre. Si ce n'est le cas pour aucune variable, on renvoie true et sinon false.

```
let est_close f lv =
  not (List.exists (fun v -> est_libre v f) lv)
```

8. La structure $\mathcal{M}$ n'est pas un modèle de F car ce n'est pas un modèle de F_1 : en effet, $1 < 1$ est faux. En revanche, en modifiant $\mathcal{M}$ de sorte à ce que R soit interprétée par $\leq$ sur les réels, on obtient un modèle $\mathcal{M}'$ pour F. En fait, n'importe quel ensemble ordonné est un modèle pour F.

9. a) Ce séquent est valide et on peut même remplacer l'implication par une équivalence et conserver ce fait. On l'a démontré en cours dans le cadre du calcul propositionnel mais rien ne change dans le cadre du calcul des prédicats. On note $\Gamma = \{F_1 \vee F_2, \neg F_1 \wedge \neg F_2\}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, F_1 \vdash \neg F_1 \wedge \neg F_2}\,\text{ax}}{\Gamma, F_1 \vdash \neg F_1}\,\text{elim-}\wedge \quad \dfrac{}{\Gamma, F_1 \vdash F_1}\,\text{ax}}{\Gamma, F_1 \vdash \bot}\,\text{elim-}\neg \quad \dfrac{\dfrac{\dfrac{}{\Gamma, F_2 \vdash \neg F_1 \wedge \neg F_2}\,\text{ax}}{\Gamma, F_2 \vdash \neg F_2}\,\text{elim-}\wedge \quad \dfrac{}{\Gamma, F_2 \vdash F_2}\,\text{ax}}{\Gamma, F_2 \vdash \bot}\,\text{elim-}\neg \quad \dfrac{}{\Gamma \vdash F_1 \vee F_2}\,\text{ax}}{\dfrac{\Gamma \vdash \bot}{\Gamma \vdash \neg(F_1 \vee F_2)}\,\text{intro-}\neg}\,\text{elim-}\vee}{\vdash \neg F_1 \wedge \neg F_2 \Rightarrow \neg(F_1 \vee F_2)}\,\text{intro-}\Rightarrow$$

b) Ce séquent n'est pas valide car il existe des ensembles ordonnés non totalement ordonnés. Par exemple, la structure dont le domaine est $\{a, b\}^\star$ et dans laquelle R est interprétée par l'ordre préfixe sur les mots satisfait F mais ne satisfait pas $\forall x \, \forall y \, R(x, y) \vee R(y, x)$ puisque a et b ne sont pas comparables.

c) Ce séquent est valide comme le montre l'arbre de preuve suivant :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{F \vdash F}\,\text{ax}}{F \vdash \forall x \, R(x, x)}\,\text{elim-}\wedge\ (2\ \text{fois})}{F \vdash R(x, x)}\,\text{elim-}\forall}{F \vdash \exists y \, R(x, y)}\,\text{intro-}\exists}{F \vdash \forall x \, \exists y \, R(x, y)}\,\text{intro-}\forall$$

L'intro-$\exists$ peut être utilisée car $R(x, x) = R(x, y)[x/y]$. L'utilisation de la règle intro-$\forall$ est licite car x n'est pas libre dans F. Cette preuve formalise le raisonnement sémantique suivant : si on a F, on a toujours $R(x, x)$ donc en particulier, il existe un y tel que $R(x, y)$, à savoir x lui même.

d) Ce séquent n'est pas valide car il existe des ordres bien fondés. Par exemple, la structure dont le domaine est $\mathbb{N}$ et dans laquelle R est interprétée par $\geq$ est un modèle de F mais pas un modèle de $\forall x \, \exists y \, (R(x, y) \wedge \neg(x = y))$. En effet, il n'y a pas d'entier naturel y tel que $0 \geq y$ et $y \neq 0$.

e) Ce séquent n'est pas valide car il existe des ensembles ordonnés non bornés. La structure $\mathcal{M}'$ définie à la question 8 est un modèle pour F mais pas pour $\exists x \, \forall y \, R(x, y)$.

Problème 2

1. Il y en a autant que le cardinal de A^n, c'est-à-dire p^n.
2. Il y a autant de mots de longueur n formés de lettres toutes distinctes qu'il n'y a d'applications injectives de $[\![1, n]\!]$ dans A, c'est-à-dire 0 si $n > \text{Card}(A) = p$ et $p(p-1)...(p-n+1) = \frac{p!}{(p-n)!}$ si $p \geq n$.
3. Si n est pair, il existe $k \in \mathbb{N}$ tel que $n = 2k$ et dans ce cas un palindrome de taille n est entièrement déterminé par ses k premières lettres : il y a donc $p^k = p^{n/2}$ palindromes de taille n paire, d'après la question 1.

   Si $n = 2k+1$ est impair, un palindrome de taille n est entièrement déterminé par ses $k+1$ premières lettres : il y a donc $p^{k+1} = p^{(n+1)/2}$ palindromes de taille n impaire.
4. Un mot de longueur n et dont deux lettres consécutives sont différentes se contruit comme suit : on choisit sa première lettre parmi les p possibles, la deuxième parmi les $(p-1)$ qui ne sont pas la première... la dernière parmi les $(p-1)$ qui ne sont pas l'avant-dernière. On en déduit que le nombre de tels mots est $p(p-1)^{n-1}$.
5. Choisissons l'emplacement des α_1 lettres a_1 : on a $\binom{n}{\alpha_1}$ possibilités. Pour chacun de ces choix, on a ensuite $\binom{n-\alpha_1}{\alpha_2}$ façons de placer les lettres a_2. On poursuit ce raisonnement jusqu'à avoir $\binom{n-\alpha_1-...-\alpha_{p-1}}{\alpha_p} = \binom{\alpha_p}{\alpha_p} = 1$

façon de placer les lettres a_p. Finalement, le cardinal de l'ensemble de mots considéré dans cette question est :

$$\binom{n}{\alpha_1}\binom{n-\alpha_1}{\alpha_2}...\binom{n-\alpha_1-...-\alpha_{p-1}}{\alpha_p} = \frac{n!}{\alpha_1!\alpha_2!...\alpha_p!}$$

6. Soit $u, v, w \in A^\star$. Le mot u est préfixe de lui même donc $u \preceq u$ est acquis. Si $u \preceq v$ et $v \preceq u$ alors il existe $x, y \in A^\star$ tels que $u = vx$ et $v = uy$. Dans ces conditions, $u = uyx$ et par régularité, $xy = \varepsilon$ donc $x = y = \varepsilon$ puis $u = v$. Enfin, si $u \preceq v$ et $v \preceq w$, il existe $x, y \in A^\star$ tels que $v = ux$ et $w = vy$ donc $w = uxy$ donc u est préfixe de w et ainsi $u \preceq w$.

   On en déduit que $\preceq$ est réflexive, antisymétrique et transitive donc c'est une relation d'ordre. Il n'est en général (il l'est si $p = 1$) pas total puisque sur l'alphabet $\{a, b\}$, a et b ne sont pas comparables. Il est en revanche bien fondé sur $A^\star$, un mot ayant un nombre fini de préfixes distincts.

7. Si $u \preceq v$, il existe $x \in A^\star$ tel que $v = ux$ et donc $|u| = |v| - |x| \leq |v|$. La réciproque est fausse dès qu'il y a au moins deux lettres dans A : comme vu à la question précédente, $|a| = |b| = 1$ mais on n'a pas $a \preceq b$.

8. Le mot ε est un (et même, le) plus petit élément de $(A^\star, \preceq)$ puisqu'il est préfixe de tout mot. Cet ensemble n'admet pas de plus grand élément : si m était un tel élément, ma avec $a \in A$ vérifierait $m \prec ma$ contredisant la maximalité de m.

9. Notons $m = m_1...m_n$ un majorant commun à $u = u_1...u_p$ et $v = v_1...v_q$. Supposons que $p \leq q$. Comme $u \preceq m$, pour tout $i \in [\![1, p]\!]$, $u_i = m_i$; comme $v \preceq m$, pour tout $i \in [\![1, q]\!]$, $v_i = m_i$, d'où on déduit que pour tout $i \in [\![1, p]\!]$, $u_i = v_i$ ce qui assure que $u \preceq v$. Si on $p \geq q$ on montre similairement que $u \succeq v$.

   Réciproquement, si u et v sont comparables, ils ont un majorant commun : v si $u \preceq v$ et u sinon.

10. a) Même preuve qu'en question 7 avec des inégalités strictes plutôt que larges.

    b) Montrons l'unicité : si $x = (x_k)_{k\in\mathbb{N}}$ et $y = (y_k)_{k\in\mathbb{N}}$ conviennent, soit $k \in \mathbb{N}$ et montrons que $x_k = y_k$. Par stricte croissance de $(|u_n|)_{n\in\mathbb{N}}$, établie en 10a, il existe $n \in \mathbb{N}$ tel que $|u_n| > k$. Pour ce n, on a par définiton de x et y : $[x]_{|u_n|} = u_n = [y]_{|u_n|}$ et en particulier, on a bien égalité des lettres x_k et y_k.

    Montrons l'existence. La partie unicité explique en fait comment construire la k-ème lettre du mot infini x, formalisons le. Soit $k \in \mathbb{N}$. L'ensemble des entiers n tels que $|u_n| > k$ est non vide par 10a : notons alors m_k son plus petit élément et définissons x_k comme étant la $(k+1)$-ème lettre de mot u_{m_k}. Il ne reste plus qu'à montrer que la suite x définie ainsi convient.

    Soit $n \in \mathbb{N}$ et $k \in [\![0, |u_n| - 1]\!]$. Alors on a $m_k \leq n$ par construction de m_k. Comme x_k est la $(k+1)$-ème lettre de u_{m_k} et que $(u_n)_{n\in\mathbb{N}}$ est croissante, x_k est aussi la $(k+1)$-ème lettre de u_n. On a donc bien $[x]_{|u_n|} = u_n$ et l'existence de la limite de u est démontrée.

11. a) Une récurrence immédiate montre que pour tout $n \in \mathbb{N}$, $|u_n| = 2^{n+1}$.

    b) La suite des tailles des mots u_n est strictement croissante d'après ce qui précède et la suite u est croissante pour l'ordre lexicographique donc u est en fait strictement croissante pour cet ordre. La question 10b montre qu'elle converge. Sa limite est le mot infini x tel que pour tout $k \in \mathbb{N}$ la lettre x_k vaut a si k est pair et b sinon.

12. a) On a $\varphi_2 = aba$, $\varphi_3 = abaab$ et $\varphi_4 = abaababa$.

    b) La suite φ est évidemment croissante vu sa définition. De plus, une récurrence immédiate montre que pour tout $n \in \mathbb{N}$, $|\varphi_n| < |\varphi_{n+1}|$ d'où on déduit que pour tout $n \in \mathbb{N}$, $\varphi_n \neq \varphi_{n+1}$ ce qui assure en fait la stricte croissance de φ. La question 10b permet donc de conclure que φ converge.

13. Si f et h sont deux morphismes de monoïdes prolongeant g, montrons qu'ils sont égaux. Soit $u \in A^\star$ dont on note $u_1, ...u_n$ les lettres. On a :

$$\begin{aligned} f(u) &= f(u_1)f(u_2)...f(u_n) \text{ par définition d'un morphisme de monoïde} \\ &= g(u_1)...g(u_n) \text{ par hypothèse sur } f \\ &= h(u_1)...h(u_n) = h(u) \end{aligned}$$

On a donc bien $f = h$. Pour l'existence, il suffit de définir f comme suit : si $u = \varepsilon$, $f(u) = \varepsilon$ et sinon, $u = u_1...u_n$ où les u_i sont des lettres de A et dans ce cas on pose $f(u) = g(u_1)g(u_2)...g(u_n)$. Il est aisé de vérifier que f est bien un morphisme convenable.

14. On obtient $\tau_1 = ab$, $\tau_2 = abba$, $\tau_3 = abbabaab$ et $\tau_4 = abbabaabbaababba$.

On montre par récurrence immédiate que pour tout $n \in \mathbb{N}$, $|\tau_n| = 2^n$. La clé de la preuve réside dans le fait que le morphisme μ double la taille du mot qu'il prend en argument puisque ce comportement est observé sur toutes les lettres de A.

15. Vu la stricte croissance de $(|\tau_n|)_{n\in\mathbb{N}}$ établie précédemment, il suffit de montrer la croissance de τ pour pouvoir conclure à sa stricte croissance. Montrons donc par récurrence sur $n \in \mathbb{N}$ que $\tau_n \preceq \tau_{n+1}$.

C'est bien le cas pour $n = 0$ puisque $\tau_0 = a \preceq ab = \tau_1$. Si ce résultat est établi pour $n \in \mathbb{N}$, il existe $y \in A^\star$ tel que $\tau_{n+1} = \tau_n y$. Comme μ est un morphisme de monoïdes, $\tau_{n+2} = \mu(\tau_{n+1}) = \mu(\tau_n)\mu(y) = \tau_{n+1}\mu(y)$ et on a donc bien $\tau_{n+1} \preceq \tau_{n+2}$ ce qui conclut la récurrence.

16. Le fait que β est un (endo) morphisme est une banalité ; la question est ici de montrer que β et μ commutent. Or,

$$\beta(\mu(a)) = \beta(ab) = ba = \mu(b) = \mu(\beta(a)) \text{ et } \beta(\mu(b)) = \beta(ba) = ab = \mu(a) = \mu(\beta(b))$$

d'où on déduit que $\beta \circ \mu$ et $\mu \circ \beta$ coïncident sur toutes les lettres de A. L'unicité prouvée à la question 13 garantit alors l'égalité de ces deux morphismes et par suite le résultat souhaité.

17. a) Montrons par récurrence sur $n \in \mathbb{N}$ que $\tau_{n+1} = \tau_n\overline{\tau_n}$. C'est éminemment vrai au rang 0. De plus, si cette propriété est vérifiée pour $n \in \mathbb{N}$, on a

$$\tau_{n+2} = \mu(\tau_{n+1}) = \mu(\tau_n\overline{\tau_n}) = \mu(\tau_n)\mu(\overline{\tau_n}) = \tau_{n+1}\overline{\mu(\tau_n)} = \tau_{n+1}\overline{\tau_{n+1}}$$

La première égalité est la définition de τ, la deuxième l'application de l'hypothèse de récurrence, la troisième l'utilisation du fait que μ est un morphisme, la quatrième résulte de la question 16 et la dernière à nouveau de la définition de τ.

On montre alors par récurrence que pour tout $n \in \mathbb{N}$, $x_n = \tau_n$ et $y_n = \overline{\tau_n}$. C'est le cas pour $n = 0$. Si c'est le cas au rang n, on a

$$x_{n+1} = x_n y_n = \tau_n\overline{\tau_n} = \tau_{n+1}$$

d'après ce qui précède d'une part, et d'autre part l'involutivité de β assure que

$$y_{n+1} = y_n x_n = \overline{\tau_n}\tau_n = \overline{\overline{\tau_n}\tau_n} = \overline{\tau_n\overline{\tau_n}} = \overline{\tau_{n+1}}$$

b) On a $|x_0|_a = 1$ et $|x_0|_b = 0$. Pour $n \geq 1$, une récurrence immédiate montre que $|x_n|_a = |x_n|_b = 2^{n-1}$.

L'hérédité s'appuie sur la relation entre x_n et τ_n établie à la question précédente, le fait que pour tout $n \in \mathbb{N}^\star$, τ_n contient autant de a que de b (résultat qui lui même se prouve... par récurrence) et le fait que la taille de τ_n est 2^n, résultat obtenu en question 14.

c) Cela se montre (encore) par récurrence sur $n \in \mathbb{N}$. C'est acquis pour $n = 0$. Si $n \in \mathbb{N}$ est tel que x_{2n} et y_{2n} sont des palindromes, $x_{2n+2} = x_{2n+1}y_{2n+1} = x_{2n}y_{2n}y_{2n}x_{2n}$ est bien un palindrome et un raisonnement similaire montre qu'il en va de même pour y_{2n+2}.

d) Si m est un mot de $A^\star$, notons ${}^t m$ le transposé du mot m. Alors, pour tout $n \in \mathbb{N}$, ${}^t x_{2n+1} = y_{2n+1}$ car

$$\begin{aligned} {}^t x_{2n+1} &= {}^t(x_{2n} y_{2n}) \text{ par définition} \\ &= {}^t y_{2n}\, {}^t x_n \\ &= y_{2n} x_{2n} \text{ puisque } x_{2n} \text{ et } y_{2n} \text{ sont des palindromes par 17.b} \\ &= y_{2n+1} \end{aligned}$$

18. D'après la question 14 et la définition de t, les 2^n premières lettres de t sont exactement les lettres de τ_n. Pour montrer le résultat, il suffit de donc de prouver par récurrence sur $n \in \mathbb{N}$ la propriété $P(n)$ suivante : "pour tout $k \in [\![0, 2^n - 1]\!]$, la lettre d'indice k dans τ_n (en commençant les indices à 0) est un a si et seulement si l'écriture binaire de k comporte un nombre pair de uns".

    La lettre d'indice 0 de $\tau_0 = a$ est un a et il n'y a aucun un dans l'écriture binaire de 0 donc $P(0)$.

    Supposons que $P(n)$ est vérifiée et considérons le mot τ_{n+1}. Pour tout $k \in [\![0, 2^n - 1]\!]$, la lettre d'indice k dans τ_{n+1} est un a si et seulement si l'écriture binaire de k comporte un nombre pair de uns d'après l'hypothèse de récurrence. Il ne reste donc plus qu'à établir le résultat souhaité pour les lettres de τ_{n+1} d'indice $k \in [\![2^n, 2^{n+1} - 1]\!]$. Soit donc un tel entier k.

    Alors il existe $k' \in [\![0, 2^n - 1]\!]$ tel que $k = 2^n + k'$. L'écriture binaire de k comporte donc un 1 de plus que celle de k'. De plus, $\tau_{n+1} = \tau_n \overline{\tau_n}$ (voir question 17.a), donc la lettre d'indice k dans τ_{n+1} est la conjuguée de la lettre d'indice k' dans τ_n. Ainsi, la lettre d'indice k dans τ_{n+1} est un a si et seulement si la lettre d'indice k' est un b dans τ_n, si et seulement si k' comporte un nombre impair de uns par hypothèse de récurrence, si et seulement si k comporte un nombre pair de uns. On a bien prouvé $P(n+1)$ ce qui conclut.

    L'écriture binaire de 100 comporte 3 occurrences de 1 (puisque c'est 1100100) ; on en déduit donc que $t_{100} = b$.

19. Commençons par (re)souligner une propriété $(\star)$ importante du morphisme μ : il double la taille des mots.

    Soit $m \in A^\star$ sans facteur cube. Supposons par l'absurde que $\mu(m)$ contient un facteur cube. Alors il existe $u, v, w \in A^\star$ tels que $u \neq \varepsilon$ et $\mu(m) = vu^3w$.

    - Si $|u|$ et $|v|$ sont toutes les deux paires, alors u et v sont les images de facteurs de x par μ (car μ est un morphisme qui vérifie $(\star)$). On peut alors écrire m sous la forme $m = v'(u')^3w'$ avec $u = \mu(u')$ et $v = \mu(v')$. Le mot u' ne peut pas être égal à ε, sans quoi son image par μ, à savoir u, le serait aussi ce qui n'est pas. On en déduit alors que m admet un facteur cube : c'est une contradiction.

    - Si $|u|$ est paire et $|v|$ est impaire, v contient au moins une lettre donc a une dernière lettre. Supposons que cette dernière lettre est un a. Alors il existe v' tel que $v = v'a$ et $\mu(m) = v'auuuw$. Comme $|v'|$ est paire, $(\star)$ assure que v' est l'image par μ d'un facteur de m et le facteur ax de $\mu(m)$, avec x la première lettre de u est donc l'image par μ d'une certaine lettre de m. Vu la définition de μ, cela garantit que la première lettre de u est un b.

      Le mot u étant de longueur paire, on peut ainsi le décomposer de la façon suivante : $u = bzl$ où z est un facteur de u de longueur paire et z est une lettre de $\{a, b\}$. Mézalors,

      $$\mu(m) = \underbrace{v'}_{\text{longueur paire}} \underbrace{ab}_{\text{longueur de 2}} \underbrace{z}_{\text{longueur paire}} \underbrace{lb}_{\text{longueur de 2}} zlbzlw$$

      et la propriété $(\star)$ nous informe que le facteur lb est l'image par μ d'une lettre, ce qui, vu la définition de μ, impose que $l = a$. On peut dès lors réécrire $\mu(m)$ comme suit :

      $$\mu(m) = v'(abz)(abz)(abz)aw$$

      et comme v' est de longueur paire et que abz aussi, on retombe dans le cas précédemment traité, pour lequel une contradiction a déjà été établie. Le cas où la dernière lettre de v est b se traite de même.

- Le dernier cas à traiter est celui où $|u|$ est impaire. Si v est de longueur paire, comme dans le point précédent, la propriété $(\star)$ nous assure que $u = u_1...u_k l$ avec les u_i des mots de deux lettres images par μ d'une des lettres de m et $l \in \{a, b\}$. Par exemple, $l = a$. On a donc

$$\mu(m) = \underbrace{v}_{\text{longueur paire}} \quad \underbrace{u_1...u_k}_{\text{longueur paire}} \quad au_1...u_k au_1...u_k aw$$

et ainsi, le mot de deux lettres commençant par a et terminant par la première lettre de u_1 est l'image par μ d'une lettre de m, ce qui vu la définition de μ assure que u_1 commence par un b. Donc la première lettre de u est un b. Comme u_1 est image d'une lettre par μ, la deuxième lettre de u est un a. Donc :

$$\mu(m) = \underbrace{\underbrace{v}_{\text{longueur paire}} \quad ba \quad \underbrace{u_2...u_k}_{\text{longueur paire}} \quad ab\, au_2...u_k aba...u_k aw}_{\text{longueur paire}}$$

Le même raisonnement que ci-dessus montre que la première lettre de u_2 est un b, donc la troisième lettre de u est un b, puis sa quatrième un a. En poursuivant récursivement, on montre ainsi que les lettres d'indice impair dans u sont des b et les lettres d'indice pair sont des a (en commençant la numérotation des indices à 1). Mais comme u est de longueur impaire, sa dernière lettre, à savoir a, est d'indice impair donc devrait être un b. C'est une contradiction ! Le cas où $l = b$ se traite de la même façon.

Si v est de longueur impaire, le raisonnement suivant s'applique toujours : on décompose simplement u en $lu_1...u_k$ avec $l \in \{a, b\}$ et les u_i des mots de deux lettres images par μ d'une lettre de m.

On aboutit ainsi à une contradiction dans tous les cas ce qui montre que $\mu(m)$ n'admet aucun facteur cube.

20. A présent, on peut facilement montrer par récurrence que pour tout $n \in \mathbb{N}$, τ_n est sans facteur cube. C'est bien le cas pour $\tau_0 = a$ et l'hérédité est immédiate vu la question 19 et la définition de τ.

Si t avait un facteur cube, il existerait un $n \in \mathbb{N}$ tel que τ_n admette un facteur cube, mais ce n'est pas possible d'après ce qui précède. On en déduit que t est sans facteur cube.

21. On constate en les listant exhaustivement que tous les mots de longueur 4 sur $\{a, b\}$ admettent un facteur carré, d'où on déduit que tous les mots de longueur supérieure à 4 sur $\{a, b\}$ admettent un facteur carré. Répondre à cette question revient donc à lister tous les mots de longueur inférieure à 3 et sans facteur carré ; on trouve : ε, a, b, ab, ba, aba et bab.

22. a) Notons L le langage dénoté par $(ab+ba)^\star$. On montre aisément que tout mot m de L vérifie $|m|_a = |m|_b$. Si $m \in L$, les mots ama et bmb ne vérifient pas cette propriété donc ne peuvent pas appartenir à L.

b) Les principes utilisés dans cette preuve sont similaires à ceux de la preuve de la question 19. Par définition de t, il suffit de montrer pour tout $n \in \mathbb{N}$, τ_n ne contient aucun facteur de la forme $lvlvl$ avec $l \in \{a, b\}$. Montrons le par récurrence.

C'est bien le cas pour τ_0. Soit donc $n \in \mathbb{N}$ vérifiant l'hypothèse de récurrence et supposons par l'absurde que τ_{n+1} admet un facteur de la forme $lvlvl$ avec $v \in \{a, b\}^\star$ et $l \in \{a, b\}$. Alors il existe $x, y \in \{a, b\}^\star$ tels que $\tau_{n+1} = xlvlvly$. On sait d'ailleurs que $v \neq \varepsilon$, sans quoi t contiendrait le facteur cube l^3 ce qui contredirait la question 20.

- Si $|x|$ et $|v|$ sont paires, la propriété $(\star)$ assure que lvl et v sont l'image par μ d'un facteur de τ_n donc appartiennent tous les deux à $(ab + ba)^\star$ ce qui est impossible d'après 22a.

- Si $|x|$ est paire et $|v|$ est impaire, v se factorise en zv' ou $z \in \{a, b\}$ et $v' \in \{a, b\}^\star$ est de longueur paire. Comme $|x|$ est paire, le mot lz est l'image d'une lettre de τ_n par μ donc $z = \bar{l}$. D'où :

$$\tau_{n+1} = \underbrace{xl\bar{l}v'l\bar{l}v'}_{\text{de longueur paire}} \quad ly$$

On en déduit que la première lettre de y est nécessairement $\bar{l}$ et finalement τ_{n+1} se factorise en

$$\tau_{n+1} = \underbrace{x}_{\text{longueur paire}} \underbrace{(l\bar{l}v'l\bar{l}v'l\bar{l})}_{\text{longueur paire}} y'$$

Ainsi, le mot $l\bar{l}v'l\bar{l}v'l\bar{l}$ est l'image par μ d'un facteur de τ_n de la forme $zwzwz$ avec $z \in \{a,b\}$ et $w \in \{a,b\}^\star$ ce qui est impossible d'après l'hypothèse de récurrence.

- Si $|x|$ est impaire et $|v|$ est paire, la propriété $(\star)$ assure à nouveau que v et lvl sont image par μ d'un facteur de τ_n et on conclut à une contradiction comme dans le premier cas.
- Si $|x|$ et $|v|$ sont impaires, la factorisation de τ_{n+1} et la propriété $(\star)$ de μ permettent de prouver que les dernières lettres de x et de v (qui existent) sont égales à $\bar{l}$ donc il existe $x', v' \in \{a,b\}^\star$ tels que $v = v'\bar{l}$ et $x = x'\bar{l}$. Dans ces conditions,

$$\tau_{n+1} = x'\bar{l}(lv')\bar{l}(lv')\bar{l}ly$$

avec x' de longueur paire et lv' de longueur impaire : on retombe donc dans le deuxième cas traité.

Dasn tous les cas on a une contradiction donc τ_{n+1} n'a aucun facteur de la forme $lvlvl$ avec $l \in \{a,b\}$ et l'hérédité est acquise.

c) Notons σ l'unique (d'après la question 13) endomorphisme de $\{a,b,c\}^\star$ tel que $\sigma(a) = abb$, $\sigma(b) = ab$ et $\sigma(c) = a$. Le mot infini θ vérifie par construction $\sigma(\theta) = t$.

Supposons par l'absurde que θ admet un facteur carré, $u \neq \varepsilon$. Le mot θ étant infini, il existe alors un facteur de la forme u^2l dans θ avec l une lettre de $\{a,b,c\}$. Mézalors, l'image par σ de ce facteur est un facteur de t. Or,

$$\sigma(u^2l) = \sigma(u)\sigma(u)\sigma(l) = au'au'az$$

avec $u', z \in \{a,b,c\}^\star$ car l'image par σ de n'importe quelle lettre commence toujours par un a et que u contient au moins une lettre vu qu'il n'est pas réduit à ε. On en déduit que $au'au'a$ est un facteur de t ce qui contredit le résultat de la question 22b. En conséquence, θ est sans facteur carré.

23. On appelle S_1 la séquence de coups de gauche et S_2 celle de droite. On considère à présent la suite de coups S définie comme suit. On lit le mot infini de Thue-Morse de gauche à droite : si un a est lu, on effectue la suite de coups S_1, si c'est un b, on effectue la suite de coups S_2. Comme le mot de Thue-Morse est sans facteur cube (question 20), S ne contient aucune séquence de coups qui se répète trois fois d'affilée.

On vient donc de trouver une suite infinie de coups licites aux échecs et que la règle de l'énoncé n'interrompt jamais. Ainsi, cette règle est insuffisante pour garantir que toute partie d'échecs termine.

Problème 3

1. Cette fonction est juste un renommage de la fonction strcmp !

```
int inferieur(char* u, char* v)
{
    return strcmp(u,v);
}
```

2. Soit $u, v \in A^\star$ tel que $u \neq v$. On peut sans perte de généralité supposer que $|u| \geq |v|$. Considérons $u_1 ... u_{k-1}$ le plus long préfixe de u qui est aussi préfixe de v (égal à ε si $k = 1$).

   Si ce préfixe est égal à u alors u est un début de v donc $u \prec v$. Sinon, la lettre u_k existe et on a $u_k \neq v_k$ par construction : si $u_k < v_k$ on a $u \prec v$ et si $u_k > v_k$, on a $v \prec u$.

   On vient de (re)démontrer que l'ordre lexicographique est total.

3. Il n'y a pas compatibilité par concaténation à droite : $0 \prec 00$ mais $01 \not\prec 001$.

4. On réserve de la place sur le tas pour $n + 1$ caractères où n est la taille du mot m en entrée afin de pouvoir stocker en dernière position le caractère \0 signifiant la fin de la chaîne qu'on renverra. Un assert permet de répondre aux exigences de l'énoncé quant à la vérification de la licéité du décalage i :

```
char* conjugue(char* m, int i)
{
    int n = strlen(m);
    assert(i >= 0 && i < n);
    char* c = (char*) malloc ((n+1)*sizeof(char));
    for (int j = 0 ; j < n-i ; j++)
    {
        c[j] = m[j+i];
    }
    for (int j = n-i ; j < n; j++)
    {
        c[j] = m[j-n+i];
    }
    c[n] = '\0';
    return c;

}
```

5. Un mot de longueur n a au moins un conjugué : lui même. Il y a atteinte dès qu'on considère un mot dont toutes les lettres sont identiques (quel que soit l'alphabet). Un mot de longueur n a au plus n conjugués et on a atteinte dès que toutes les lettres du mot sont différentes (ce qui arrive dès que A dispose de plus de n lettres).

6. La relation $\mathcal{C}$ est évidemment réflexive, symétrique et transitive.

7. Dans chacun des cas, on calcule la liste L des conjugués du mot u considéré et on vérifie que u est le plus petit mot de L selon l'ordre lexicographique.

   a) 0010011 est un mot de Lyndon.

   b) 010011 n'est pas un mot de Lyndon car 001101 en est un conjugué strictement plus petit.

c) 001001 n'est pas un mot de Lyndon car faire un décalage de 3 lettres vers la gauche de ce mot en donne un conjugué non trivial qui est égal à 001001 et pas strictement plus grand.

8. Les mots de une lettre n'ont aucun conjugué non trivial : les merveilleuses propriétés de l'ensemble vide assurent donc qu'un tel mot est un mot de Lyndon.

9. Il suffit de comparer le mot en entrée à tous ses conjugués non triviaux. Lorsqu'on est en présence d'un mot de taille un, on ne passe jamais dans la boucle et la valeur renvoyée par lyndon est cohérente avec la réponse à la question 6 :

```
bool lyndon(char* m)
{
    int n = strlen(m);
    bool res = true;
    for (int i = 1; i < n; i++)
    //on commence à 1 pour éviter le conjugué trivial
    {
        char* c = conjugue(m, i);
        res = res && (inferieur(m,c) < 0);
        free(c);
    }
    return res;
}
```

10. On utilise la stratégie suivante : on calcule la classe d'équivalence du premier (selon l'ordre lexicographique) mot de longueur 4 dont on ignore encore dans quelle classe il se trouve. On obtient 6 classes d'équivalences :

$$\{0000\}, \{0001, 0010, 0100, 1000\}, \{0011, 0110, 1100, 1001\}, \{0101, 1010\}, \{0111, 1110, 1101, 1011\}, \{1111\}$$

11. D'après l'énoncé, pour trouver les mots de Lyndon de taille 4, il suffit de déterminer les colliers apériodiques parmi ceux ci-dessus puis de trouver le plus petit élément selon l'ordre lexicographique dans chacun.

    Les colliers apériodiques sont les deuxième, troisième et cinquième. Dans chacun, le plus petit élément est en fait le premier élément. On en déduit qu'il y a 3 mots de Lyndon de taille 4 : $0001, 0011$ et 0111.

12. On concatène 00111 à lui même pour obtenir un mot de taille 9 (il est nécessaire pour ce faire de tronquer la dernière lettre) ; on obtient le mot 001110011. On supprime tous les 1 en fin de ce mot ce qui donne 0011100. On remplace enfin le 0 final par un 1 et ainsi le mot ajouté à E dans ces conditions sera 0011101.

13. On obtient dans cet ordre les mots $0, 0001, 001, 0011, 01, 011, 0111, 1$. En observant les mots de taille 4 trouvés via cet algorithme, on constate que cette réponse est cohérente avec celle de la question 10.

14. Il s'agit d'expliquer pourquoi la boucle tant que extérieure termine. C'est le cas car m croît strictement (pour l'ordre lexicographique) au fil des itérations de cette boucle et a toujours une taille bornée par n. Comme l'ensemble des mots de taille n sur $\{0, 1\}$ est borné par 2^n, il arrive un moment ou m n'est constitué que de 1, et alors la boucle tant que intérieure supprime toutes les lettres de m : à l'itération suivante, $m = \varepsilon$.

15. Les instructions de la boucle de la ligne 4 ont une complexité pire cas en $O(n)$: on fait au pire (et en fait ce cas n'arrive jamais) n ajouts de lettres aux lignes 5-6 puis n suppressions de lettres à la ligne 9. Le raisonnement ci-dessus montre par ailleurs qu'on passe dans cette boucle au maximum 2^n fois. On obtient donc une complexité en $O(n2^n)$.

16. On obtient la factorisation suivante : $(1)(01)(001011)(001)(0)(0)$.

17. La propriété suivante : "les mots $u^{(1)}, ..., u^{(k)}$ de la factorisation courante sont tous des mots de Lyndon" est un invariant. C'est initialement vrai puisque les mots d'une lettre sont des mots de Lyndon (cf question 8).

    Si ça l'était avant une itération, cela le reste après puisque la nouvelle factorisation n'a fait que fusionner deux mots (par hypothèse) de Lyndon u, v consécutifs et vérifiant $u \prec v$. L'énoncé assure que dans ces conditions uv reste un mot de Lyndon et donc tous les mots de la nouvelle factorisation sont de Lyndon.

    En particulier, suite à la dernière itération tous les mots de la factorisation $(u^{(1)}, ..., u^{(k)})$ sont des mots de Lyndon et ces derniers vérifient $u^{(1)} \succeq ... \succeq u^{(k)}$ d'après la condition d'arrêt des itérations de l'algorithme $\mathcal{A}$. Cette factorisation est par définition une factorisation de Lyndon ce qui conclut.

18. Aucune difficulté si ce n'est qu'il faut penser à nouveau à terminer correctement la sous-chaîne.

```c
char* extrait_chaine(char* mot, int debut, int fin)
{
    int taille_sous_chaine = fin-debut+2;
    char* sous_chaine = (char*) malloc(taille_sous_chaine*sizeof(char));
    for (int i = 0; i < taille_sous_chaine; i++)
    {
        sous_chaine[i] = mot[debut +i];
    }
    sous_chaine[fin-debut+1] = '\0';
    return sous_chaine;
}
```

19. C'est la fonction non triviale de l'énoncé. On détaille un peu la conception de cet algorithme.

    L'idée générale est de stocker en permanence deux entiers indiquant où se trouvent la fin du prochain mot à lire dans la factorisation et la fin du successeur de ce mot dans la factorisation. C'est le rôle des variables fin_premier_mot et fin_deuxieme_mot.

    Connaître ces deux entiers permet d'extraire les deux prochains facteurs — premier_mot et deuxieme_mot — à étudier dans la factorisation. Une fois extraits à l'aide de extrait_chaine, on compare ces derniers à l'aide de inferieur. S'ils sont correctement triés, on passe au couple de facteurs suivants en mettant à jour fin_premier_mot, fin_deuxieme_mot et par conséquent premier_mot et deuxieme_mot. Sinon, on a trouvé un défaut de tri : c'est l'entier stocké dans fin_premier_mot !

    Si après vérification de tous les couples de facteurs consécutifs, on constate qu'aucun ne viole la décroissance de la factorisation, on peut conclure que la factorisation donnée est correctement triée. Dans tous les cas, on n'oubliera pas de faire les libérations de mémoire nécessaires.

    La première boucle tant que (qui termine car facto contient au moins un true si elle représente la factorisation d'un mot non vide) et la première conditionnelle servent juste à récupérer la fin du tout premier premier mot et à s'interrompre au cas où la factorisation donnée ne contient en fait qu'un seul mot.

```
bool est_trie(char* mot, bool* facto, int* defaut_de_tri)
{
    bool res = true;
    int n = strlen(mot);
    int fin_premier_mot = 0;
    while (facto[fin_premier_mot] == false)
    {
        fin_premier_mot++ ;
    }
    if (fin_premier_mot == (n-1))
    {
        return true;
    }
    else
    {
        char* premier_mot = extrait_chaine(mot, 0, fin_premier_mot);
        int fin_deuxieme_mot = fin_premier_mot +1;
        for (int i = fin_deuxieme_mot ; i < n ; i++)
        {
            if (facto[i])
            {
                fin_deuxieme_mot = i;
                char* deuxieme_mot = extrait_chaine(mot, fin_premier_mot+1,␣
↪fin_deuxieme_mot);
                if (inferieur(premier_mot, deuxieme_mot) < 0)
                {
                    *defaut_de_tri = fin_premier_mot;
                    free(premier_mot);
                    free(deuxieme_mot);
                    return false;
                }
                else
                {
                    free(premier_mot);
                    premier_mot = deuxieme_mot;
                    fin_premier_mot = fin_deuxieme_mot;
                    fin_deuxieme_mot = fin_deuxieme_mot + 1;
                }
            }
        }
        free(premier_mot);
        return true;
    }
}
```

20. Le gros du tavail est déjà fait ! On initialise une factorisation fin_de_mots selon les modalités de l'énoncé avec un tableau contenant true dans chaque case. Tant que cette factorisation n'est pas triée (ce qui se vérifie avec est_trie), on récupère l'indice k de fin d'un mot plus petit que son successeur et on fusionne ce mot avec son successeur en faisant simplement basculer la valeur de la case k à false dans la factorisation.

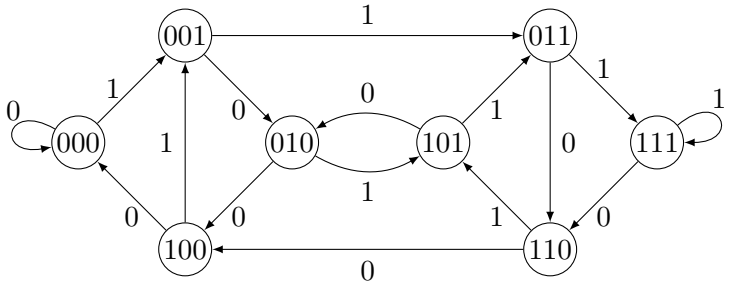
```
bool* factorisation(char* mot)
{
    int n = strlen(mot);
    bool* fin_de_mots = (bool*) malloc(n*sizeof(bool));
    for (int i = 0; i <n ; i++)
    {
        fin_de_mots[i] = true;
    }

    int defaut_de_tri  = (-1);
    while (!(est_trie(mot,fin_de_mots,&defaut_de_tri)))
    {
        fin_de_mots[defaut_de_tri] = false;
    }
    return fin_de_mots;
}
```

21. Le mot *abba* est un mot de de Bruijn d'ordre 2 sur $\{a, b\}$.

22. La longueur d'un mot de de Bruijn d'ordre n sur un alphabet de taille k est k^n. En effet, il y a bijection entre les positions des lettres d'un mot de de Bruijn et l'ensemble des mots de taille n sur A par construction.

23. On obtient le graphe (planaire !) suivant :



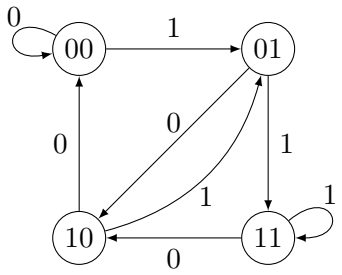
24. Soit s un sommet de $B(k, n)$. Par construction d'un graphe de de Bruijn, il existe $a \in A$ et $m \in A^\star$ tel que (l'étiquette de) s soit am et la fonction :

$$\begin{cases} A \to \{\text{arêtes sortant de } am\} \\ b \mapsto (am, mb, b) \end{cases}$$

est une bijection claire, d'où on déduit qu'il y a autant d'arêtes sortant de s que d'éléments de A, à savoir k. Le même raisonnement s'applique pour les arêtes entrant sur s en remarquant que s peut aussi s'écrire mb avec m un mot de $A^\star$ et b une lettre de A. Ainsi, les degrés entrant et sortant de s valent k.

25. Il suffit de compter le nombre d'arcs sortants de $B(n, k)$ pour conclure. Ce graphe a k^n sommets, chacun de ses sommets est de degré sortant k d'après 24 et on en déduit donc que $B(k, n)$ comporte k^{n+1} arcs.

26. La méthode est la même que pour la question 23 :



Le chemin $(01, 10, 00, 00, 01, 11, 11, 10, 01)$ est un circuit eulérien dans $B(2, 2)$. L'étiquette de ce chemin est 00011101. Ce mot est un mot de de Bruijn d'ordre 3 (ce que la suite de l'énoncé confirme).

27. Soit $n \in \mathbb{N}^\star$ et $k \in \mathbb{N}$. Le graphe $B(k, n)$ est connexe : si $u = u_1...u_m$ et $v = v_1...v_l$ sont deux sommets, le chemin $(u_0...u_m, u_1...u_m v_1, u_2...u_m v_1 v_2, ..., u_m v_1...v_{l-1}, v_1...v_l)$ relie u et v. De plus la question 24 montre que tout sommet de $B(k, n)$ voit son degré entrant et son degré sortant être égaux. Ainsi, $B(k, n)$ contient un circuit eulérien et comme ce dernier est un mot de de Bruijn d'ordre $n + 1$ d'après l'énoncé, on a :

    Pour tout $n \geq 2$, pour tout alphabet A de taille k, il existe un mot de de Brujin sur A d'ordre n.

28. Pour répondre à cette question, il faut d'abord calculer les mots de Lyndon de taille 4, 2 et 1. Mais ceci a déjà été fait en question 13 : les mots convoités sont $0, 0001, 0011, 01, 0111, 1$ dans l'ordre lexicographique. En les concaténant, on en déduit que 0000100110101111 est le plus petit mot de de Bruijn d'ordre 4.

29. Il y a k^n codes possibles, chacun de longueur n. A priori, il faudrait donc entre $n \times k^n$ chiffres dans le digicode.

30. Plutôt que d'entrer successivement tous les codes, on entre un mot de de Brujin sur l'alphabet $[\![0, k-1]\!]$ d'ordre n. Ce dernier contient toutes les séquences de n chiffres de $[\![0, k-1]\!]$ ce qui garantit l'ouverture de la porte et ne nécessite de taper "que" k^n chiffres d'après la question 22.